CS 740, Fall 1997

Assignment 5:

Part I: Adding Compression and Encryption to TCP

Assigned: Mon., Nov. 3
Due: Mon., Nov. 17

Policy

You may work in a group of up to 3 people in solving the problems for this assignment. You should turn in a single report for your entire group, identifying all of the group members.

Logistics

Any clarifications and revisions to the assignment will be posted on the class bboard and Web page. There is no electronic handin for this assignment.

1. The impact of encryption and compression on TCP performance

In this part of the assignment you will measure the rate at which TCP can transfer data across computer networks. You will then assess the costs and benefits of encrypting and compressing the data before transferring it.

The files you need for this part of the assignment are available in the directory:

`/afs/cs.cmu.edu/academic/class/15740-f97/public/asst/asst5/`

We have provided a suite of working programs to help you with this part of the assignment. These programs are broken into two groups.

1. `ttcp.c`
2. `pgp262si.tar`

The first group consists of a single program, `ttcp.c`. This program is used to test the performance of TCP. You can read more about it in file `README.ttcp`.

The second group of files is the installation package for the Pretty Good Privacy (PGP) encryption scheme. I have prepared a condensed version of the document `setup.doc` in file `short_setup.doc`. This file provides instructions for installing PGP. Installation was quite straightforward on my SPARCstation 4 running Solaris.

Note: Please do not distribute the PGP files. As far as I can tell, it is perfectly legal to use PGP within the United States, but it may not be legal to export the files to foreign countries. (Nevertheless, the files are easily obtained over the internet.)

Task 1: Evaluating the performance of TCP

In order to evaluate the impact of adding encryption and data compression to the TCP protocol, we will begin by analyzing the performance of "plain old" TCP using the `ttcp` program.

You are to perform three experiments.

1. Measure the rate at which streams of synthetic data can be transferred using TCP when both the sender and receiver are running on the same physical machine, e.g., on your workstation.
2. Measure the rate at which data can be transferred when the sender and receiver are running on machines connected to the same ethernet. For this experiment, try to isolate the machines from all other ethernet traffic. If this is not possible, please perform your experiments at a time when network usage is minimal, so as not to interfere with other users.
3. Measure the rate at which TCP data can be transferred between two machines located on different subnetworks. In particular, between one machine located on the CMU subnetwork 128.2.0.0, and the other outside of CMU. Report the locations of the two endpoints. Better yet – provide a trace of the path through the internet from the source to the destination. Please do this experiment an off-peak hour.

Task 2: Evaluating the benefit of compression

As we discussed in class, today's modems compress can uncompress data on-the-fly, often yielding an improvement of a factor of 2 in effective bandwidth. For this task, we will experiment with adding on-the-fly data compression and uncompression to TCP running over ethernet.

As in Task 1, you will use `ttcp` to evaluate the performance of TCP. In that task, the `ttcp` program most likely generated and then transferred synthetic data. In this task, however, you will feed an input stream into `ttcp`, which it will then transfer. This input stream will be generated by one of the standard Unix compression programs such as `compress`, `zip`, or `gzip` (your choice). Your compression program will read its input from a file.

Feeding the output of a compression program into `ttcp` may require some tinkering. You'll have to figure out how to ensure that `ttcp` accounts for any time spent doing the compression.

For your experiments, you will use three different types of input files:

1. highly compressible data, e.g., all zeros,
2. moderately compressible data, e.g., a text file or program,
3. uncompressible data, e.g., random data.

Note that if you generate your uncompressible data by compressing some other form of data (rather than using random data), you run the risk that the compression program will recognize the data as already compressed (by looking at the file header) and not perform any compression.

For each of the three types of data, you are to repeat the experiments of Task 1. Namely:

1. establish a connection between two ports on the same machine,
2. between two ports on different machines on the same ethernet, and

3. between machines separated by some distance.

Please report the make and speed of your CPU as well. Does compression improve the overall transfer rate in any of these experiments?

Task 3: Evaluating the cost of encryption

Your last task is to evaluate the cost of encrypting the data stream using the Pretty Good Privacy encryption scheme. To complete this portion of the assignment, you will have to install PGP on your machine. Read the directions in file `short_setup.doc`.

Repeat the experiments from Task 2, this time using `pgp` rather than a compression program to feed `ttcp`. You should perform two types of experiments,

1. running `pgp` with compression turned off, and
2. running `pgp` with both compression and encryption.

Thus, for Task 3 you'll have a total of $3 * 3 * 2 = 18$ little experiments to perform: 3 types of network connections, 3 types of input files, and 2 modes of running `pgp`. Note that by default `pgp` compresses a file before it encrypts it. To disable compression, you might have to change one or two lines in the file `pgp.c`. Note also that `pgp` attempts to detect if a file has already been compressed, in which case it won't try to compress it.